Discounted Duration Calculus*

Heinrich Ody[1]**, Martin Fränzle[1]***, and Michael R. Hansen[2]†

[1] Department of Computing Science, University of Oldenburg, Germany
fraenzle@informatik.uni-oldenburg.de, heinrich.ody@uni-oldenburg.de
[2] DTU Compute, Technical University of Denmark mire@dtu.dk

Abstract. To formally reason about the temporal quality of systems discounting was introduced to CTL and LTL. However, these logic are discrete and they cannot express duration properties. In this work we introduce discounting for a variant of Duration Calculus. We prove decidability of model checking for a useful fragment of discounted Duration Calculus formulas on timed automata under mild assumptions. Further, we provide an extensive example to show the usefulness of the fragment.

Keywords. duration calculus, temporal logic, model checking, timed automata, discounting

1 Introduction

In economics discounting represents that money earned soon can be reinvested earlier and hence yields more revenue than money earned later. Discounting has been introduced into temporal logics to represent that something happening earlier is more important than similar events happening later [14]. A typical example is a rail-road crossing. Consider the property "eventually the gates are open". While a controller leaving the gates closed an hour after the train has passed might be safe and alive, it is not useful. We can use discounting to express that the controller should not wait unnecessarily long before opening the gates. The discount here is a scalar defining the decrease rate of an exponential function assigning weights to events based on their (relative) time of occurrence. In [1, 13, 14] such weighted evaluation of temporal properties has been described as quantifying the temporal quality of a system.

Duration Calculus (DC) [12] was introduced to reason about *duration properties* of real time systems. In the prominent gas burner case study [24] the

* An early version of this work was presented at the Nordic Workshop of Programming Theory 2015.

** Work of the author is supported by the Deutsche Forschungsgemeinschaft (DFG) within the Research Training Group DFG GRK 1765 SCARE.

*** Work of the author was partially supported by Deutsche Forschungsgemeinschaft within the Transregional Collaborative Research Center SFB/TR 14 AVACS.

† Work of the author was partially supported by the Danish Research Foundation for Basic Research within the IDEA4CPS project.

following duration property was proven: "in any time interval of length ≥ 60 gas is leaking for at most 5% of the time". The great expressiveness of DC however, makes automated reasoning in most cases undecidable [10, 11, 15].

So far discounting in logics only has been studied for discrete-time temporal logics (LTL, CTL*, μ-calculus) [1, 13, 14, 20, 21]. Here, we study discounting in the dense-time logic DC. Our interest in DC arises from its expressiveness, being able to express properties of accumulated durations instead of just temporal distances, and the consequential undecidability of most fragments over dense time. A primary objective of the work reported herein thus is to investigate the impact of discounting on effective approximability of model checking for DC formulas.

To this end we define *discounted Duration Calculus* (DDC), where the truth value is real-valued in the interval $[0, 1]$, instead of Boolean. A truth value closer to 1 means *higher temporal quality*. We point out that we use exponential discounting because this is the most common from of discounting. However, other discounting mechanisms are possible. With DDC we can express properties such as $\Diamond^d \phi$ (meaning "*soon* with *discount* d the system satisfies ϕ"), where ϕ is a DDC formula and $\Diamond$ is the *right neighbourhood* modality from [9]. To evaluate the truth value of $\Diamond^d \phi$ on the interval $[t_0, t_1]$ we search for a neighbouring interval $[t_1, t_2]$ such that the discounting factor $d^{t_2 - t_1}$ multiplied with the truth value of ϕ on $[t_1, t_2]$ is maximal.

Our main result is that for the fragment $\mathrm{DDC}_{<1}$, which consists of all DDC formulas where all discounts are < 1, model checking is approximable. This stems from the fact that the effect of the system behaviour on the satisfaction value becomes negligible as time advances. Hence, for approximation it suffices to only consider bounded prefixes of runs, which in turn enables us to use bounded model checking. Our model-checking method is extended to cope with modalities of the form $\mathrm{G}_S \, \phi$ (meaning "whenever S happens ϕ holds thereon"). We provide an extensive example illustrating the usefulness of our approach.

Related Work. Discounting in temporal logics was first studied in [14] and later in [1, 13, 20, 21]. However, in all of these works the logics are discrete and they cannot express duration properties. In [7] the authors introduce a method to perform model checking on weighted (or priced) timed automata with weighted versions of CTL and LTL. A cost in their work essentially corresponds to the duration of a state variable in our work. However, they do not consider discounting and in their case model checking becomes undecidable for automata with at least three clocks. For a fragment without duration properties called *test formulas*, which are used to express undesired behaviours, model checking has been shown decidable [22]. In [17] the authors define a model checking procedure for a fragment that allows duration properties, but disallows negation of the chop operator. In [16] the authors give a real-valued interpretation to DC and they provide an approximative procedure to check satisfiability. However, the authors do not consider model checking. Further, in none of the works on DC discounting is considered.

2 Discounted Duration Calculus (DDC)

We use an adapted version of Duration Calculus (DC), where the chop operator is replaced by a right neighbourhood modality. As atomic formulas, we allow comparison of linear combinations of durations with constants.

Definition 1 (Syntax of DDC). *Let* $d, k_0, \ldots, k_n, c \in \mathbb{Q}$, *where* $d \in [0,1]$, $\gtrsim \in \{\geq, >\}$ *and let* $P \in AP$ *denote arbitrary* atomic propositions *(or state variables or just variables). Then the formulas* ϕ *of* Discounted Duration Calculus *(abbreviated* DDC*) and* state expressions S *are defined by the grammar*

$$\phi ::= \Diamond^d \phi \mid \neg\phi \mid \phi \vee \phi \mid \Sigma_{i=0}^n k_i \textstyle\int S_i \gtrsim c \ ,$$
$$S ::= P \mid \neg S \mid S \vee S \ .$$

We denote the fragment of DDC where all discounts are < 1 *as* $DDC_{<1}$.

Let AP be a finite set of atomic propositions. The semantics of DC is defined in terms of timed words. A *timed word* is a (possibly infinite) sequence

$$\tau = (\sigma_0, t_0)(\sigma_1, t_1) \cdots (\sigma_i, t_i) \cdots$$

where $\sigma_i \in 2^{AP}$ and $t_0 = 0$ and $t_i \in \mathbb{R}_{\geq 0}$. The sequence of time stamps $t_0, t_1, \ldots$ occurring in τ must be *weakly monotonically increasing*, that is $t_i \leq t_{i+1}$. Furthermore, we require *progress* in infinite timed words τ, that is, for every $t \in R_{\geq 0}$ there is an $i > 0$ such that $t_i > t$.

If τ is an infinite sequence, then we say that the *time span (or just span)* of τ comprises the non-negative reals and we write $\text{span}(\tau) = \mathbb{R}_{\geq 0}$. If τ is a finite sequence having (σ_n, t_n) as its last element, the span of τ is the bounded (right-open) interval $[0, t_n)$ and we write $\text{span}(\tau) = [0, t_n)$. We shall from now on restrict our attention to timed words having a non-empty time span.

For a timed word $\tau = (\sigma_0, t_0)(\sigma_1, t_1) \cdots (\sigma_i, t_i) \cdots$ and $\delta \in \text{span}(\tau)$, where $\delta > 0$, we define the *time-bounded prefix* τ_δ of τ as the timed word:

$$\tau_\delta = (\sigma_0, t_0)(\sigma_1, t_1) \cdots (\sigma_i, t_i)(\sigma_i, \delta)$$

where i is given by $t_i \leq \delta < t_{i+1}$. Note that there is exactly one such i since $\delta \in \text{span}(\tau)$.

A timed word $\tau = (\sigma_0, t_0)(\sigma_1, t_1) \cdots (\sigma_i, t_i) \cdots$ induces a function

$$\tau(P) : \text{span}(\tau) \to \{0, 1\}$$

for every atomic proposition P, as follows:

$$\tau(P)(t) = \begin{cases} 1 & \text{if } P \in \sigma_i, \text{ for some } i \text{ where } t_i \leq t < t_{i+1}, \\ 0 & \text{otherwise}. \end{cases}$$

The function $\tau(P)$ is also called a *trajectory* for P. Trajectories are lifted to state expressions by a point-wise extension in a straightforward manner, for example, $\tau(S_0 \vee S_1)(t) = \tau(S_0)(t) \vee \tau(S_1)(t)$. We use the abbreviation S_τ for $\tau(S)$. Notice

that the progress requirement for infinite timed words guarantees that for every variable P, every finite part of P_τ has a finite number of discontinuity points, i.e. P_τ is of *finite variability*.

The *semantics of a DDC formula* ϕ on the basis of a timed word $\tau = (\sigma_0, t_0)(\sigma_1, t_1)\cdots(\sigma_i, t_i)\cdots$ is a function:

$$\tau(\phi) : \mathit{Intv} \to [0, 1]$$

where $\mathit{Intv} = \{[t, t'] \subseteq \mathrm{span}(\tau) \mid t \leq t'\}$ denotes the set of bounded and closed real intervals contained in $\mathrm{span}(\tau)$. The function assigns to $\tau(\phi)\,[a, b]$ a *satisfaction value* in the real interval $[0, 1]$, where closer to 1 means better.

Discounts d occur only in connection with the right neighbourhood modality $\Diamond^d\phi$, which expresses that an adjacent interval to the right of the current interval satisfies ϕ. The discount d is used to decrease the satisfaction value as the length of the adjacent interval necessary to find a satisfaction of ϕ increases. The modal formula $\Diamond^d\phi$ can be understood as "*soon* ϕ holds".

Definition 2 (Semantics of DDC). *The* semantics of a formula*, given a timed word* τ *and an interval* $[t_0, t_1]$*, is defined as*

$$\begin{aligned}
\tau(\Diamond^d\phi)\,[t_0, t_1] &= \sup\{d^{t_2 - t_1} \cdot \tau(\phi)\,[t_1, t_2] \mid t_2 \geq t_1 \wedge t_2 \in \mathrm{span}(\tau)\} \\
\tau(\Sigma_{i=0}^n k_i \textstyle\int S_i \gtrsim c)\,[t_0, t_1] &= \begin{cases} 1 & \text{if} \Sigma_{i=0}^n k_i \int_{t=t_0}^{t_1} \tau(S_i)(t)\,\mathrm{d}t \gtrsim c \\ 0 & \text{otherwise} \end{cases} \\
\tau(\neg\phi)\,[t_0, t_1] &= 1 - \tau(\phi)\,[t_0, t_1] \\
\tau(\phi_0 \vee \phi_1)\,[t_0, t_1] &= \max\{\tau(\phi_0)\,[t_0, t_1], \tau(\phi_1)\,[t_0, t_1]\}
\end{aligned}$$

where $\gtrsim \in \{>, \geq\}$.

If we want to use the standard neighbourhood modalities without discounting then we use a discount of 1. In this case we do not explicitly write the discount.

We define as abbreviation a modality

$$\Box^d\phi = \neg\Diamond^d\neg\phi\,,$$

which can be understood as "ϕ holds for a *long* time". For some interval $[t_0, t_1]$ the semantics is

$$\tau(\Box^d\phi)\,[t_0, t_1] = 1 - \sup\{d^{t_2 - t_1} \cdot (1 - \tau(\phi)\,[t_1, t_2] \mid t_2 \geq t_1 \wedge t_2 \in \mathrm{span}(\tau))\}\ .$$

We point out that the supremum searches for a small $t_2 \geq t_1$ that makes the truth value of $\tau(\phi)$ on $[t_1, t_2]$ small. Further, the greater the interval $[t_1, t_2]$ is chosen, the greater the truth value of $\Box^d\phi$ becomes. Note that the truth value of $\Box^d\phi$ increases with the decrease of d, while the truth value of $\Diamond^d\phi$ decreases with the decrease of d.

To express that a state expression S holds throughout an interval, we use the abbreviation:

$$\lceil S \rceil = \textstyle\int \neg S = 0 \wedge \ell > 0$$

where ℓ is an abbreviation of $\int(S' \vee \neg S')$ for an arbitrary state expression S'.

With $\Diamond\,\Diamond\,\phi$ we express that on some right interval, which may or may not be adjacent to the current interval, ϕ holds. We shall use the abbreviation $\mathrm{F}_S\,\phi$ to denote that there is some future point interval, say $[t,t]$, where S "happens" and ϕ holds, that is, ϕ holds on $[t,t]$ and S changes from 0 to 1 at t and keeps the value 1 for some nonzero time:

$$\mathrm{F}_S\,\phi = \Diamond\,\Diamond\left(\lceil\neg S\rceil \wedge \begin{pmatrix} \Diamond\lceil S\rceil \\ \wedge\,\Diamond\,(\ell = 0 \wedge \phi)\end{pmatrix}\right)$$

Let $\mathrm{G}_S\,\phi = \neg\mathrm{F}_S\,\neg\phi$. The formula $\mathrm{G}_S\,\phi$ thus means that for all future time points t, if S happens at t, then ϕ holds on $[t,t]$.

Example 1. As an example we consider the three formulas:

$$\begin{aligned}
\phi_0 &= \Diamond^{\,0.8}(\textstyle\int P \geq 3)\\
\phi_1 &= \Diamond^{\,0.9}\,\Box^{\,0.8}(\textstyle\int P - \int\neg P \leq 3)\\
\phi_2 &= \mathrm{G}_Q\,\Diamond^{\,0.8}(\textstyle\int P \geq 2)
\end{aligned}$$

and the two timed words:

- $\tau_0 = (\{P\},0)\ (\{\},2)\ (\{P\},3)\ (\{\},5)\ (\{P\},6)$
  $(\{\},8)\ (\{P\},9)\ (\{\},11)\ (\{P\},12)(\{\},14),$

- $\tau_1 = (\{\},0)\ (\{Q\},1)\ (\{P\},2)\ (\{Q\},4)\ (\{P,Q\},5)$
  $(\{\},6)\ (\{P\},7)\ (\{\},8)\ (\{Q\},9)\ (\{\},10)$
  $(\{P\},11)\ (\{\},12)\ (\{P\},13)\ (\{\},14),$

which induce the trajectories depicted in Fig. 1.

These above formulas can be explained as follows:

- ϕ_0 reads "soon P has held for at least 3 time units",
- ϕ_1 reads "soon P should hold no more than 3 time units more than $\neg P$, for a long time", and
- ϕ_2 reads "every time Q changes its value from 0 to 1, then soon P has held for 2 time units".

Evaluate ϕ_0 on τ_0: The earliest point when $\int P \geq 3$ is satisfied is at $t = 4$. We calculate:

$$\begin{aligned}
&\tau_0(\Diamond^{\,0.8}(\textstyle\int P \geq 3))\,[0,0]\\
&= \sup\{0.8^t \cdot \tau_0(\textstyle\int P \geq 3)\,[0,t] \mid t \in \mathrm{span}(\tau_0)\}\\
&= 0.8^4 \approx 0.41
\end{aligned}$$

Evaluate ϕ_1 on τ_0: In ϕ_1 the inner modality is given t_0 and chooses the smallest t_1 such that $\int P - \int \neg P \leq 3$ is violated. The outer modality chooses t_0 such that the product of its discount 0.9^{t_0} multiplied with the truth value archived by the inner modality becomes maximal. We calculate (assuming that $t_0, t_1 \in \text{span}(\tau_0)$):

$$\begin{aligned}
&\tau_0(\Diamond^{0.9} \Box^{0.8}(\textstyle\int P - \int \neg P \leq 3))\,[0,0] \\
&= \sup_{t_0 \geq 0}\{0.9^{t_0} \cdot (1 - \sup_{t_1 \geq t_0}\{0.8^{t_1 - t_0} \cdot (1 - \tau_0(\textstyle\int P - \int \neg P \leq 3)\,[t_0, t_1])\})\} \\
&= 0.9^2 \cdot (1 - 0.8^{12-2} \cdot (1 - \tau_0(\textstyle\int P - \int \neg P \leq 3)\,[t_0, t_1]) \\
&= 0.9^2 \cdot (1 - 0.8^{12-2} \cdot (1 - 0)) \approx 0.72
\end{aligned}$$

Evaluate ϕ_2 on τ_1: We evaluate $\psi = \Diamond^{0.8}(\int P \geq 2)$ on all point intervals $[t, t]$, where Q changes its value from 0 to 1. For τ_1 these points are $1, 4$ and 9. The truth value is $\min\{\tau_1(\psi)\,[1,1], \tau_1(\psi)\,[4,4], \tau_1(\psi)\,[9,9]\}$, which evaluates to $\min\{0.8^{4-1}, 0.8^{8-4}, 0.8^{14-9}\} \approx 0.33$.

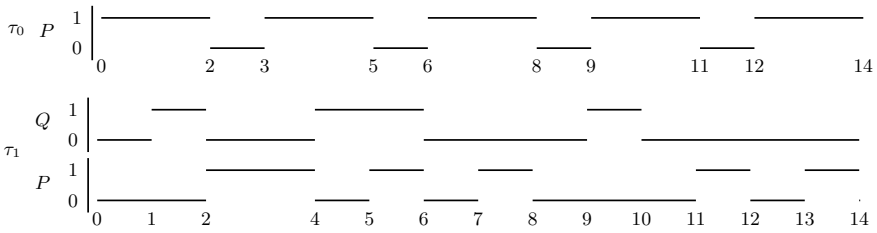

Fig. 1. Graphical representation of two timed words. The word τ_1 contains two atomic propositions P and Q. We assume that all values remain 0 after time point 14.

3 Model Checking

In this section we prove that model checking for a relevent fragment of DDC is approximable, where the model is given as a timed automaton [2]. To this end we first show that for approximation it is sufficient to consider only bounded prefixes of runs. Then we give a reduction to quantified linear real arithmetic.

3.1 The Model

As model we use timed automata that have atomic propositions that hold in states (denoted by Λ) instead of events on edges. Additionally, our timed automata have a set of allowed initial clock valuations, where the initial value of a clock may be different from 0. Further, we assume that our timed automata are *strongly non-Zeno* [3]. This is the case, iff there is a non-zero constant $c \in \mathbb{R}_{>0}$ such that in every control cycle at least c units of time passes. Formally, for every path $l_0 \xrightarrow{e_0} \ldots \xrightarrow{e_{n-1}} l_n$ with $l_0 = l_n$ there is an edge that resets some clock x and an edge or a location with a constraint $x \geq c$. For ease of exposition we assume this constant to be a natural number greater than 0.

Definition 3 (Timed Automata). *Let $\mathcal{X}$ be a finite set of non-negative real-valued variables, called clocks and let $\mathbb{V}$ be the set of all clock valuations. Then $\mathbb{B}(\mathcal{X})$ is the set of all conjunctions of constraints of the form $x - y \bowtie c$ or $x \bowtie c$ with $x, y \in \mathcal{X}, c \in \mathbb{Q}, \bowtie \in \{<, >, \geq, \leq\}$. Further, let AP be a finite set of atomic propositions. A timed automaton is a tuple $A = (L, E, I, Inv, \Lambda, \mathcal{X})$, where L is the set of locations, $E \subseteq L \times \mathbb{B}(\mathcal{X}) \times 2^{\mathcal{X}} \times L$ is the set of edges, $I \subseteq L \times \mathbb{V}$ is the set of initial states, $Inv : L \to \mathbb{B}(\mathcal{X})$ are the invariants per location and $\Lambda : L \to \mathbb{B}(AP)$ assigns a set of atomic propositions which hold in a location.*

Note that commonly I is defined as $L' \times \{\mathbf{0}\}$, where $L' \subseteq L$ and $\mathbf{0} \in \mathbb{V}$ is the clock valuation where all clocks have value 0.

Let ν be a clock valuation, R a set of clocks and g a guard. We define $\nu + t$ is the clock valuation where the values of all clocks are increased by t. With $\nu[R \mapsto 0]$ we denote the clock valuation resulting from ν by setting all clock values in R to 0. And with $\nu \in g$ we denote that ν satisfies the constraints in g.

Definition 4 (Runs of Timed Automata). *Given a timed automaton $A = (L, E, I, Inv, \Lambda, \mathcal{X})$ and a possibly infinite timed word $\tau = (\sigma_0, t_0) \dots (\sigma_i, t_i) \dots$ let $\Delta_i = t_{i+1} - t_i$ and let N be the set of integers such that $i \in N$ iff there is an element (σ_i, t_i) in τ. This means that $N = \mathbb{N}$ if τ is infinite. A run of A on τ is a sequence*

$$\pi = (l_0, \nu_0) \dots (l_i, \nu_i) \dots$$

with $(l_0, \nu_0) \in I$, for every $j \in N$ we have $\sigma_j \implies \Lambda(l_j)$ and for every $j, j{+}1 \in N$ there exists an edge $(l_j, g_j, R_j, l_{j+1}) \in E$ such that $\forall t \in \mathbb{R}. 0 \leq t \leq \Delta_j \implies \nu_j + t \in Inv(l_j)$, $\nu_j + \Delta_j \in g$, $\nu_{j+1} \in Inv(l_{j+1})$ and $\nu_{j+1} = (\nu_j + \Delta_j)[R_j \mapsto 0]$.

With $L(A)$ we denote the set of all timed words for which there exists a run on A.

As we work with real-valued truth values, here model checking gives a value in the interval $[0, 1]$.

Definition 5 (Model Checking Timed Automata). *Let A be a timed automaton and ϕ be a DDC formula. We define* model checking *as computing*

$$\min_{\tau \in L(A)} \{\tau(\phi)\,[0,0]\}\,.$$

When the timed automaton has upper bounds for the values of all clocks in all locations the set of reachable states is computable with a finite representation. The goal of this constraint is to avoid over approximation introduced by the *normalisation* step of reachability algorithms [5]. We use this to reduce computing the satisfaction value of $\mathrm{G}_S\,\phi$ by A to computing the satisfaction value of ϕ by a transformed automaton A'.

Lemma 1. *Let ϕ be a $DDC_{<1}$ formula, $A = (L, E, I, Inv, \Lambda, \mathcal{X})$ a timed automaton with $\forall l \in L, x \in \mathcal{X}. \exists c \in \mathbb{Q}. x \lesssim c \in Inv(l), \lesssim \in \{<, \leq\}$, and S a state expression. Then*

$$\min_{\tau \in L(A)} \{\tau(\mathrm{G}_S\,\phi)\,[0,0]\} = \min_{\tau' \in L(A')} \{\tau'(\phi)\,[0,0]\}$$

where $A' = (L, E, I', Inv, \Lambda, \mathcal{X})$ *is the timed automaton obtained from* A*, by letting the initial states* I' *be those where the state expression* S *just has become true. Let Reach be the set of reachable states in* A*, let* L_S *be the set of locations where* S *holds and define*

$$I' = \{(l,\nu) \mid l \in L_S \wedge (l', g, R, l) \in E \wedge (l', \nu') \in Reach \\ \wedge \nu' \in g \wedge \nu \in Inv(l) \wedge \nu'[R \mapsto 0] = \nu \wedge l' \in L \setminus L_S\}\,.$$

Furthermore, I' *is computable and has a finite representation using linear arithmetic [5].*

We give our definition of approximate model checking.

Definition 6 (Approximate Model Checking). *Let* A *be a timed automaton,* ϕ *be a* $DDC_{<1}$ *formula and let* $\epsilon \in (0,1]$ *be the desired precision. Then* approximate model checking *is to compute a truth value* $v \in \mathbb{R}$ *with* $0 \leq v \leq 1$ *such that*

$$v \in \min_{\tau \in L(A)} \{\tau(\phi)\,[0,0]\} \pm \epsilon\,.$$

For this we compute the point in time $\delta = \log_d \epsilon$ such that the value of v is almost not affected by any suffix of the timed word starting at time δ. This is possible because all modalities in $\mathrm{DDC}_{<1}$ are discounted by less than 1 and hence the effect of a timed word on the truth value becomes less and less as time advances. Note that for other discounting functions, e.g. $\frac{1}{1+d\cdot(t-t')}$ other computations are necessary. However, for any computable strictly monotonic discounting function with limit 0 such a point, after which the effect on the truth value is $\leq \epsilon$, is computable.

Lemma 2. *Given a* $DDC_{<1}$ *formula* ϕ *and an allowed error* ϵ*, let* d_{m} *be the largest discount constant occurring in* ϕ *such that for all other discounts* d *in* ϕ *we have* $d \leq d_{\mathrm{m}}$ *and let* $\delta = \log_{d_{\mathrm{m}}} \epsilon$*. Then for any timed word* τ *we have*

$$|\tau(\phi)\,[0,0] - \tau_\delta(\phi)\,[0,0]| \leq \epsilon\,.$$

We transform the approximate model checking problem for $\mathrm{DDC}_{<1}$ to quantified linear real arithmetic, which we now define.

Definition 7 (Quantified Linear Real Arithmetic (QLRA)). *We define the syntax of* quantified linear real arithmetic *(QLRA) as*

$$\begin{aligned} \phi &::= \neg\phi \mid \phi \vee \phi \mid term \lesssim term \mid \exists x.\, \phi\,, \\ term &::= a \mid term + term \mid a \cdot term \mid x\,, \\ a &::\in \mathbb{Q} \end{aligned}$$

where $\lesssim \in \{<, \leq\}$ *and* x *is a variable over* $\mathbb{R}$*.*

With linear arithmetic *we denote the fragment of QLRA where all quantifiers are located under an even number of negations.*

To check to what extent a timed automaton satisfies a formula we use bounded reachability checking via linear arithmetic. The following lemma specifies which variables we use in the bounded reachability checking encoding. The construction can be found, e.g., in [4, 25].

Lemma 3 (Bounded Reachability, e.g. [4, 25]). *Given a timed automaton A, an initial zone and a step bound l, we can encode the existence of a run of length $\leq l$, starting at any state in the initial zone, in linear arithmetic. We shall assume that this run is described using variables $\underline{t_i}, \underline{P_i}$, for $0 \leq i \leq l$, describing whether in the interval $[t_i, t_{i+1})$ the propositional variable P holds or not.*

3.2 Encoding of the Semantics for Formulas

We encode the semantics of DDC in QLRA. As the semantics of DDC uses exponentials we cannot encode the exact semantics. However, we can approximate the truth value with finite but arbitrary high precision. We use this encoding to prove that approximative model checking for strongly non-Zeno timed automata is computable.

Suppose that $F(\bar{y})$ is a formula of QLRA having $\bar{y}$ as free variables (and possibly others) and suppose that $e(\bar{y})$ is a linear term, then we can express

$$x = \text{lub}\{e(\bar{y}) \in \mathbb{R} \mid F(\bar{y})\}$$

in QLRA, using the abbreviations:

$$\begin{aligned}\text{UB}(x, e(\bar{y}), F(\bar{y})) &= \forall \bar{y}.F(\bar{y}) \implies x \geq e(\bar{y}) \\ \text{LUB}(x, e(\bar{y}), F(\bar{y})) &= \text{UB}(x, e(\bar{y}), F(\bar{y})) \wedge \forall z.\text{UB}(z, e(\bar{y}), F(\bar{y})) \implies z \geq x\end{aligned}$$

Furthermore, we shall use the following QLRA abbreviation to express that $x = \max(e_1, e_2)$:

$$\text{MAX}(x, e_1, e_2) = (e_1 < e_2 \implies x = e_2) \wedge (e_1 \geq e_2 \implies x = e_1)$$

When v, t, d range over a bounded domain we can approximate an exponential function $v \cdot d^t$ with an arbitrary precision using linear approximations. Below we will use the abbreviation x isApproxOf $v\ d\ t$ to denote that x is an approximation of $v \cdot d^t$.

The encoding of a formula ϕ in an interval $[t_0, t_1]$ is based on a symbolic first-order formula representation of a bounded model guaranteed by Lemma 3. We shall now show how the semantics of formulas on bounded runs are encoded in QLRA, by defining a QLRA formula x isSemOf^l $\phi\ t_0\ t_1$ denoting that x is (an approximation of) the semantics of ϕ in the interval $[t_0, t_1]$. This formula is defined by recursion over the structure of ϕ.

Encoding for $\boldsymbol{\tau(\Sigma_{j=0}^{n} k_j \int S_j \gtrsim c)\,[t_0, t_1]}$

We show the encoding of $k \int S \gtrsim c$. The generalisation to linear combinations of durations is easily done in QLRA.

For every interval from $\underline{t_i}$ to $\underline{t_{i+1}}$ we introduce a variable x_i denoting the duration of S on this interval. To this end we introduce the following abbreviations:

z isOverlap$_i$ t_0 t_1 denotes that z is the length of $[t_0, t_1] \cap [\underline{t_i}, \underline{t_{i+1}}]$ and
y isDur$_i$ S t_0 t_1 denotes that y is the duration of S on $[t_0, t_1] \cap [\underline{t_i}, \underline{t_{i+1}}]$.

where the definitions are provided below.

For the formula x isSemOf^l $k \int S \gtrsim c$ t_0 t_1 we define that if the inequality $(k \int S \gtrsim c)$ holds $x = 1$, and otherwise $x = 0$:

$$\big((\exists y_0, \ldots, y_{l-1}.\, k \cdot \Sigma_{i=0}^{l-1} y_i \gtrsim c \wedge \bigwedge_{i=0}^{l-1} (y_i \text{ isDur}_i\ S\ t_0\ t_1)) \implies x = 1\big) \wedge$$
$$\big(\neg(\exists y_0, \ldots, y_{l-1}.\, k \cdot \Sigma_{i=0}^{l-1} y_i \gtrsim c \wedge \bigwedge_{i=0}^{l-1} (y_i \text{ isDur}_i\ S\ t_0\ t_1)) \implies x = 0\big)$$

It is easy to generalize this to cover linear sums of accumulated durations.

The abbreviation z isOverlap$_i$ t_0 t_1 is as follows:

$$\begin{aligned}
&(t_0 \geq \underline{t_{i+1}} \vee t_1 \leq \underline{t_i} \implies z = 0)\\
\wedge\ &(t_0 \leq \underline{t_i} \wedge \underline{t_{i+1}} \leq t_1 \implies z = \underline{t_{i+1}} - \underline{t_i})\\
\wedge\ &(\underline{t_i} \leq t_0 \wedge \underline{t_{i+1}} \leq t_1 \implies z = \underline{t_{i+1}} - t_0)\\
\wedge\ &(t_0 \leq \underline{t_i} \wedge t_1 \leq \underline{t_{i+1}} \implies z = t_1 - \underline{t_i})\\
\wedge\ &(\underline{t_i} \leq t_0 \wedge t_1 \leq \underline{t_{i+1}} \implies z = t_1 - t_0)
\end{aligned}$$

and the abbreviation y isDur$_i$ S t_0 t_1 is:

$$\begin{aligned}
&(\underline{S} \implies y \text{ isOverlap}_i\ t_0\ t_1)\\
\wedge\ &(\neg \underline{S} \implies y = 0)
\end{aligned}$$

where $\underline{S}$ is the formula obtained from S by replacing every occurrence of a state variable P with $\underline{P_i}$.

Encoding of $\tau(\phi_0 \vee \phi_1)\ [t_0, t_1]$

The formula x isSemOf^l $(\phi_0 \vee \phi_1)$ t_0 t_1 is defined by:

$$\exists y_0, y_1.\, (y_0 \text{ isSemOf}^l\ \phi_0\ t_0\ t_1) \wedge (y_1 \text{ isSemOf}^l\ \phi_1\ t_0\ t_1) \wedge \text{MAX}(x, y_0, y_1)$$

Encoding of $\tau(\neg\phi)\ [t_0, t_1]$

The formula x isSemOf^l $(\neg\phi)$ t_0 t_1 is defined by $\exists y.\, (y \text{ isSemOf}^l\ \phi\ t_0\ t_1) \wedge x = 1 - y$

Encoding of $\tau(\Diamond\, {}^{d}\phi)\,[t_0, t_1]$

The formula x isSemOf^{l} $(\Diamond\, {}^{d}\phi)$ t_0 t_1 is defined by $\exists t_2, r.\, \mathrm{LUB}(x, e(y), F(t_2, y, r))$, where

$$\begin{array}{ll} e(y) & = y \\ F(t_2, y, r) & = (r \text{ isSemOf}^{l} \phi\ t_1\ t_2)\ \wedge (y \text{ isApproxOf } r\ d\ (t_2 - t_1)) \\ & \quad \wedge t_2 \geq t_1 \wedge t_2 \leq \underline{t_l} \end{array}$$

We use our approximation of the semantics in QLRA and the bounded model checking approach to prove that approximate model checking is computable.

Theorem 1 (Approximate Model Checking). *Given a strongly non-Zeno timed automaton A and a DDC$_{<1}$ formula ϕ and a desired precision $\epsilon \in \mathbb{R}_{>0}$, the approximate model-checking problem is effectively computable: There is a procedure computing $v \in [0,1]$ such that*

$$v \in \min_{\tau \in L(A)} \{\tau(\phi)\,[0,0]\} \pm \epsilon\,.$$

Proof. Let $\epsilon_1, \epsilon_2 > 0$ be such that $\epsilon_1 + \epsilon_2 = \epsilon$. According to Lemma 2, we can bound the time horizon of interest to $\delta = \log_{d_\mathrm{m}} \epsilon_1$ with d_m again being the largest discount constant occurring in ϕ, thereby obtaining

$$|\tau(\phi)\,[0,0] - \tau_\delta(\phi)\,[0,0]| \leq \epsilon_1\,. \tag{1}$$

As A is strongly non-Zeno, the number of transitions occurring in A within δ time units is bounded by a constant $l \in \mathbb{N}$, which can be computed as $\lceil M\delta \rceil$ with M being the length of the longest cycle in the transition graph of A.

Given this bound l on the length of the runs to be considered, we can easily obtain (Q)LRA encodings of both the runs of A of the appropriate length $\leq l$ and of the l-bounded DDC semantics: Let

$$R_j = F_A^j(\underline{t}, \underline{P})\,,$$

where $F_A^j(\underline{t}, \underline{P})$ is the LRA-encoding of the runs of A of length j according to Lemma 3, and let

$$\mathit{Sem}_j(y) = (y \text{ isSemOf}^{j}\ \phi\ 0\ 0)\,,$$

where y isSemOf^{j} ϕ 0 0 is the above encoding of the DDC semantics, with the look-up tables for approximating exponentials being developed to accuracy ϵ_2 over the argument range $[0, \delta]$.

We furthermore introduce an abbreviation $\mathrm{GLB}(x, y, F(y))$ for a formula defining $x = \mathrm{glb}\ \{y \mid F(y)\}$ just as we did for the least upper bound. Then the satisfying valuation of $\mathrm{GLB}(x, y, \bigvee_{j=1}^{l}(R_j \wedge \mathit{Sem}_j(y)))$, which can be determined effectively by QLRA solving, satisfies

$$|x - \min_{\tau \in L(A)} \{\tau_\delta(\phi)\,[0,0]\}| \leq \epsilon_2$$

due to the accuracy of approximating the exponentials, which together with Eq. (1) in turn implies

$$\begin{aligned} & |x - \min_{\tau \in L(A)} \{\tau(\phi)\,[0,0]\}| \leq \epsilon_2 + \epsilon_1 = \epsilon \\ \iff & x \in \min_{\tau \in L(A)} \{\tau_\delta(\phi)\,[0,0]\} \pm \epsilon\,. \end{aligned}$$

□

4 Example

To support our claims that we can reason about interesting problems with DDC we provide an example in this section.

4.1 Production Cell

We consider two drilling machines that generate heat while drilling. These machines independently of each other process work pieces of different sizes, and the drilling time needed to finish a work piece depends on the size of the piece. If a machine drills for a long time without interruption the machine becomes too hot. If the machine is too hot, it will gradually take damage. It is undesirable to always avoid that the machine becomes too hot, because then production will be too low. The desired property is that the machine soon cools down, after it became too hot.

Let $i \in \{0, 1\}$. We represent that machine i is too hot by a propositional variable H_i, that the machine is drilling by D_i and the durability of the machine by the discount (here 0.9, where closer to 1 means more durable). Further, there are coefficients (here $1, 2$) representing how quickly the temperature changes over time in the respective locations and here 5 is the desired cooldown to achieve after the machine has become too hot. We formalise the desired property as

$$\mathrm{G}_{H_0}(\Diamond^{0.9}(\textstyle\int \neg D_0 - 2\int D_0 \geq 5)) \wedge \mathrm{G}_{H_1}(\Diamond^{0.9}(\textstyle\int \neg D_1 - 2\int D_1 \geq 5))\,.$$

The controllers A_0 and A_1 of the machines are depicted on the left hand side of Fig. 2. On the right hand side of Fig. 2 we depict the automaton B that determines how quickly the working pieces may appear and that assigns the working pieces nondeterministically to the machines.

4.2 Computing the Satisfaction Value

Here we focus on the satisfaction value of the subformula $\mathrm{G}_{H_0}(\Diamond^{0.9}(\int \neg D_0 - 2\int D_0 \geq 5))$. However, the satisfaction value for the other subformula is equal.

Let $C = (A_0 \parallel A_1 \parallel B) = (L, E, I, Inv, \Lambda, \mathcal{X})$ be the parallel composition of A_0, A_1 and B. To approximate the satisfaction value of $\mathrm{G}_{H_0}(\Diamond^{0.9}(\int \neg D_0 - 2\int D_0 \geq 5))$ by C we apply Lemma 1 for the first subformula and create $C' =$

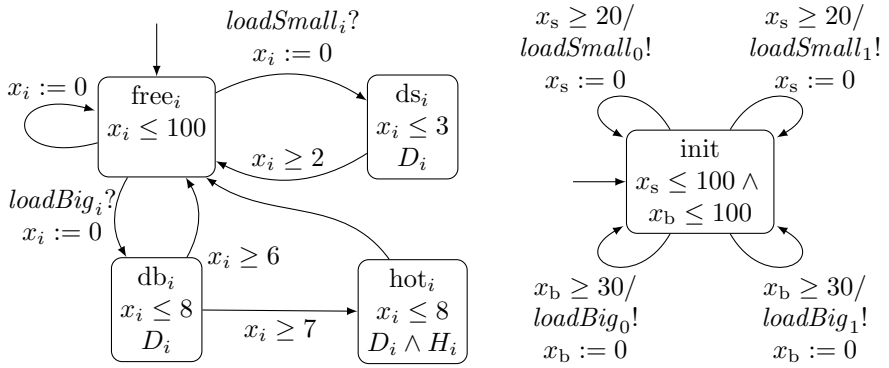

Fig. 2. On the left hand side we see the controller A_i with $i \in \{0, 1\}$ of a drilling machine. The upper bounds of 100, serve to make Lemma 1 applicable, the other upper bounds restrict the maximal drilling time needed for small and big working pieces. The self loop in free_i serves to make the parallel composition $A_0 \parallel A_1 \parallel B$ deadlock free. On the right hand side we see B, which controls how quickly work pieces may appear.

$(L, E, I', Inv, \Lambda, \mathcal{X})$ that has all states as initial states in which the edge from db_0 to hot_0 was just taken. The set I' is defined as [3]

$$\begin{aligned} I' = &\{((\text{db}_0, \text{free}_1, \text{init}), \nu) \mid \nu \in \big(7 \leq x_0 \leq 8 \wedge x_0 = x_\text{b} \wedge x_\text{s} \leq 100 \wedge \\ &\qquad ((x_1 - x_\text{s} \leq -2 \wedge x_1 \leq x_\text{b}) \vee (x_\text{b} - x_\text{s} \leq -2 \wedge x_1 \leq 100))\big)\} \cup \\ &\{((\text{db}_0, \text{ds}_1, \text{init}), \nu) \mid \nu \in (7 \leq x_0 \leq 8 \wedge x_0 = x_\text{b} \wedge x_\text{s} \leq 3 \wedge x_1 = x_\text{s})\}\,. \end{aligned}$$

Let the desired precision be $\epsilon = 0.1$. According to Lemma 2 we have $\delta = \log_{0.9} \epsilon$, which is less than 22. Let $\delta = 22$ and note that by choosing a larger δ than necessary we increase the precision of the computation. The approximation of the satisfaction value is

$$\min_{\tau \in L(C')} \{\sup\{0.9^t \cdot \tau_{22}(\textstyle\int \neg D_0 - 2\int D_0 \geq 5)\,[0,t] \mid 0 \leq t \leq 22\}\} \pm \epsilon\,.$$

Hence, we are looking for a run π in C', such that in the timed word induced by π the smallest t for which $\tau_{22}(\int \neg D_0 - 2\int D_0 \geq 5)\,[0,t]$ holds, is large.

A run that maximises the time t needed to satisfy $\int \neg D_0 - 2\int D_0 \geq 5$ is depicted below. The intuition of the run is that directly after the the machine finished a big working piece, it has to work on a small working piece. The location of B always is init. Hence, the states in the run have the form $(l_0, l_1, \nu(x_0), \nu(x_1), \nu(x_\text{s}), \nu(x_\text{b}))$ where l_i is a location from A_i with $i \in \{0, 1\}$

[3] We computed the initial states with Uppaal Tiga [8] by computing a winning strategy for the property *control* : $A[]$ *true* with the options `-c -w 2 -n 2`.

and $\nu(y)$ is the value of the clock y under the clock valuation ν:

$$\begin{aligned}\pi =&(\mathrm{hot}_0, \mathrm{free}_1, 7, 0, 19, 7)(\mathrm{hot}_0, \mathrm{free}_1, 8, 1, 20, 8)(\mathrm{free}_0, \mathrm{free}_1, 8, 1, 20, 8)\\ &(\mathrm{ds}_0, \mathrm{free}_1, 0, 1, 0, 8)(\mathrm{ds}_0, \mathrm{free}_1, 3, 4, 3, 11)\\ &(\mathrm{free}_0, \mathrm{free}_1, 3, 4, 3, 11)(\mathrm{free}_0, \mathrm{free}_1, 16, 17, 16, 24)\end{aligned}$$

The run spends 4 time units in locations where D_0 holds (hot_0, ds_0), and 13 time units in locations where $\neg D_0$ holds (free_0). Hence, it satisfies $\int \neg D_0 - 2\int D_0 \geq 5$ after $t = 17$ time units and we have

$$\min_{\tau \in L(C')} \{\tau(\Diamond^{\,0.9}(\int \neg D_0 - 2\int D_0 \geq 5))\} \in\ 0.9^{17} \pm 0.1 \approx 0.17 \pm 0.1\,.$$

In general, by considering only bounded prefixes of all runs we introduce an error of at most ϵ. However, in our example the result 0.9^{17} is exact, because in C' all runs starting from I' satisfy $\int \neg D_0 - 2\int D_0 \geq 5$ in less than $\delta = 22$ time units.

We see that the controllers of the drilling machines satisfy our cooldown property poorly. To fix this we could introduce a scheduler in between the controllers A_0, A_1 and the spawner of the working pieces B. This scheduler would then assign the working pieces to machines in a way that avoids assigning two successive working pieces to the same machine. As the model then would be quite big, we would need automation to compute the satisfaction value for the larger example. Fortunately, for strongly non-Zeno timed automata and properties of the form $\Diamond^{\,d} \Sigma_{i=0}^{n-1} k_i \int S_i \sim c$ and $\Box^{\,d} \Sigma_{i=0}^{n-1} k_i \int S_i \sim c$ we can compute the satisfaction value via optimisation modulo theories [6, 23].

5 Conclusion

Discounting has been introduced to temporal logics to formalise reasoning about temporal quality of systems [1, 13], where temporal quality quantifies how soon events of interest happen, rather than just answering the qualitative question whether they happen at all. We introduced discounting to Duration Calculus to be able to analyse the quality of real-time systems w.r.t. duration properties. Our main result is that, with the fragment $\mathrm{DDC}_{<1}$ consisting of all formulas where the discounts are < 1, we identified a fragment of DDC where model checking for timed automata is approximable under mild assumptions. While this only allows us to reason about bounded prefixes of runs, our reduction of approximating the satisfaction value for formulas $\mathrm{G}_S\,\phi$ (read: "whenever S happens ϕ holds thereon") to model checking $\mathrm{DDC}_{<1}$ enables us to also reason about infinite runs. At last, we provided an extensive example to demonstrate the usefulness of discounting in temporal logics in general and of discounting duration properties in particular.

For future work it is interesting to see how large the fragment of DDC is, for which model checking is approximable.

In Section 4 we mentioned that for properties of the form $\Diamond^{\,d} \Sigma_{i=0}^{n-1} k_i \int S_i \sim c$ and $\Box^{\,d} \Sigma_{i=0}^{n-1} k_i \int S_i \sim c$ with $d < 1$ the satisfaction value can be approximated

efficiently via a reduction to optimisation modulo theories [6, 23]. Naturally, it is desirable to find efficient algorithms for larger fragments of DDC.

Further, in [1, 13] operators, such as taking the average of two formulas, that are not available in qualitative logics, were studied. To find or define such operators and evaluate their usefulness and their effect on computability is another interesting challenge. One such operator may be $\phi \to \psi = \min\{1, 1-u+v\}$ from Łukasiewicz logics [18], where u, v are the truth values of ϕ, ψ. This definition of implication allows for a closer connection between the truth values of ϕ and ψ than the definitions we used.

Durations in our setting correspond to costs in the setting of multi-priced timed automata (MPTA) [19]. In our work we discovered that often we are interested in the costs of handling a temporal event (as indicated by our use of $\mathrm{G}_S\,\phi$). This could be modelled in MPTA by resetting the cost variable. As this reset action would not depend on the costs, but only on observable behaviour these enhanced MPTA might have interesting decidable problems.

Acknowledgement We thank Peter Gjøl Jensen for advice on how to compute the set of reachable zones with UPPAAL TIGA.

References

[1] Almagor, S., Boker, U., Kupferman, O.: Discounting in LTL. In: Tools and Algorithms for the Construction and Analysis of Systems, pp. 424–439. Springer (2014)

[2] Alur, R., Dill, D.L.: A Theory of Timed Automata. Theoretical Computer Science 126(2), 183–235 (1994)

[3] Asarin, E., Maler, O., Pnueli, A., Sifakis, J.: Controller Synthesis For Timed Automata. Symposium on System Structure and Control pp. 469–474 (1998)

[4] Badban, B., Lange, M.: Exact Incremental Analysis of Timed Automata with an SMT-Solver. In: Fahrenberg, U., Tripakis, S. (eds.) Formal Modeling and Analysis of Timed Systems. Lecture Notes in Computer Science, vol. 6919, pp. 177–192. Springer (2011)

[5] Bengtsson, J., Yi, W.: On Clock Difference Constraints and Termination in Reachability Analysis of Timed Automata. In: Dong, J.S., Woodcock, J. (eds.) International Conference on Formal Engineering Methods. Lecture Notes in Computer Science, vol. 2885, pp. 491–503. Springer (2003)

[6] Bjørner, N., Phan, A., Fleckenstein, L.: νz - an optimizing SMT solver. In: Baier, C., Tinelli, C. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 9035, pp. 194–199. Springer (2015)

[7] Bouyer, P., Larsen, K.G., Markey, N.: Model-checking one-clock priced timed automata. In: Foundations of Software Science and Computational Structures, pp. 108–122. Springer (2007)

[8] Cassez, F., David, A., Fleury, E., Larsen, K.G., Lime, D.: Efficient On-the-Fly Algorithms for the Analysis of Timed Games. In: Abadi, M., de

Alfaro, L. (eds.) Concurrency Theory. Lecture Notes in Computer Science, vol. 3653, pp. 66–80. Springer (2005)

[9] Chaochen, Z., Hansen, M.R.: An Adequate First Order Interval Logic. In: de Roever, W.P., Langmaack, H., Pnueli, A. (eds.) Compositionality: The Significant Difference. Lecture Notes in Computer Science, vol. 1536, pp. 584–608. Springer (1997)

[10] Chaochen, Z., Hansen, M.R.: Duration Calculus - A Formal Approach to Real-Time Systems. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2004)

[11] Chaochen, Z., Hansen, M.R., Sestoft, P.: Decidability and undecidability results for duration calculus. In: Enjalbert, P., Finkel, A., Wagner, K. (eds.) Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 665, pp. 58–68. Springer (1993)

[12] Chaochen, Z., Hoare, C.A.R., Ravn, A.P.: A calculus of durations. Information processing letters 40(5), 269–276 (1991)

[13] de Alfaro, L., Faella, M., Henzinger, T.A., Majumdar, R., Stoelinga, M.: Model checking discounted temporal properties. Theoretical Computer Science 345(1), 139–170 (2005)

[14] de Alfaro, L., Henzinger, T.A., Majumdar, R.: Discounting the Future in Systems Theory. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 2719, pp. 1022–1037. Springer (2003)

[15] Fränzle, M.: Model-checking dense-time Duration Calculus. Formal Aspects of Computing 16(2), 121–139 (2004)

[16] Fränzle, M., Hansen, M.R.: A robust interpretation of duration calculus. In: Dang Van Hung, Martin Wirsing (eds.) Theoretical Aspects of Computing. Lecture Notes in Computer Science, vol. 3722, pp. 257–271. Springer (2005)

[17] Fränzle, M., Hansen, M.R.: Deciding an Interval Logic with Accumulated Durations. In: Grumberg, O., Huth, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 4424, pp. 201–215. Springer (2007)

[18] Gottwald, S.: Many-valued logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. The Metaphysics Research Lab of Stanford University (2015), `http://plato.stanford.edu/archives/spr2015/entries/logic-manyvalued/`

[19] Larsen, K.G., Rasmussen, J.I.: Optimal reachability for multi-priced timed automata. Theoretical Computer Science 390(2-3), 197–213 (2008)

[20] Mandrali, E.: Weighted LTL with Discounting. In: Moreira, N., Reis, R. (eds.) Implementation and Application of Automata. Lecture Notes in Computer Science, vol. 7381, pp. 353–360. Springer (2012)

[21] Mandrali, E., Rahonis, G.: On weighted first-order logics with discounting. Acta Informatica 51(2), 61–106 (2014)

[22] Meyer, R., Faber, J., Hoenicke, J., Rybalchenko, A.: Model checking Duration Calculus: a practical approach. Formal Aspects of Computing 20(4-5), 481–505 (2008)

[23] Nieuwenhuis, R., Oliveras, A.: On SAT Modulo Theories and Optimization Problems. In: Biere, A., Gomes, C.P. (eds.) Theory and Applications of Satisfiability Testing. Lecture Notes in Computer Science, vol. 4121, pp. 156–169. Springer (2006)

[24] Ravn, A.P., Rischel, H., Hansen, K.M.: Specifying and Verifying Requirements of Real-Time Systems. IEEE Transactions on Software Engineering 19(1), 41–55 (1993)

[25] Torre, S.L., Mukhopadhyay, S., Murano, A.: Optimal-Reachability and Control for Acyclic Weighted Timed Automata. In: Baeza-Yates, R.A., Montanari, U., Santoro, N. (eds.) IFIP International Conference on Theoretical Computer Science. IFIP Conference Proceedings, vol. 223, pp. 485–497. Kluwer (2002)